第一章

类型：程序所处理的数据都保存在变量中，而每个变量都有自己的类型

内置类型：语言自身定义的类型（而形如string等类型都是标准库定义的）

[main](http://www.cnblogs.com/xzxl/p/7619716.html)的返回值：0表示成功，非0指出错误类型

从命令行运行编译器

[for语句](http://www.cnblogs.com/xzxl/p/7619862.html)

术语表：缓冲区、cerr、clog、表达式

第一部分 C++基础

第二章 变量和基本类型

几种字符类型：[char 、wchar\_t 、char16\_t 、char32\_t](http://www.cnblogs.com/xzxl/p/7645251.html" \t "_blank)

内置类型的机器实现：[内置类型如何在内存存放](http://www.cnblogs.com/xzxl/p/7645360.html" \t "_blank)

将负数转换为无符号类型：结果为无符号数的模加上这个负数

字面值常量：每个[字面值常量](http://www.cnblogs.com/xzxl/p/7645640.html" \t "_blank)都对应一种数据类型、字符串字面值（实际是由常量字符构成的数组），可以指定字面值的类型

转义序列：回车\r、换行\n

对象：指一块能存储数据并具有某种类型的内存空间

初始化：列表[初始化](http://www.cnblogs.com/xzxl/p/7764314.html" \t "_blank)、默认初始化

[变量声明和定义](http://www.cnblogs.com/xzxl/p/7645787.html)的关系：分离式编译、extern

复合类型：基于其他类型定义的类型

声明符：声明语句int &r;中，&r为声明符，声明符命名了一个变量，也指定该变量为与基本数据类型（此例为int）有关的某种类型

[引用](http://www.cnblogs.com/xzxl/p/7647156.html)&引用类型：我们称r为引用或引用类型

[指针](http://www.cnblogs.com/xzxl/p/7647306.html)：\*p为声明符，p是变量名，我们称p为指针或指针变量

[const](http://www.cnblogs.com/xzxl/p/7647816.html)限定符：常量引用（对常量的引用）、可以将一个常量引用绑定到变量、字面值、表达式（因为不允许通过该引用修改这些对象）

顶/底层const：执行对象拷贝时的限制

constexpr：声明常量

decltype类型指示符：返回表达式的类型

编写自己的头文件：头文件通常包含那些只能定义一次的实体，如类、const、constexpr变量

预处理变量：#ifdef、#ifndef、#endif

术语表：常量表达式、头文件保护符、未定义、void\*

第三章 字符串、向量和数组

标准库类型string

string::size\_type类型：string的size函数返回的是一个string::size\_type类型的值；string类及其他大多数标准库类型都定义了几种配套的类型，体现了标准库类型与机器无关的特性，类型size\_type即是其中的一种，通过域操作符来表明名字size\_type是在类string中定义的、

处理string对象中的字符：cctyoe头文件中的函数

迭代器：使迭代器失效的操作

数组：数组的引用，数组的下标类型为size\_t，数组名和指针

标准库函数begin和end

c\_str函数：返回一个C风格的字符串，返回结果是一个指针，该指针指向一个以空字符结束的字符数组，指针的类型是const char\*

使用数组初始化vector对象：vector<int> vec(begin(arr), end(arr));

术语表：直接初始化、值初始化、->运算符

第四章 表达式

表达式：最小的计算单元。一个表达式包含一个或多个运算对象，通常还包含一个或多个运算符。表达式求值会产生一个结果。

字面值和变量是最简单的表达式，其结果就是字面值和变量的值。

表达式本身也可以作为运算对象。

[左值和右值](http://www.cnblogs.com/xzxl/p/7652612.html)

运算符的优先级和结合律

[位运算符](http://www.cnblogs.com/xzxl/p/7654801.html)：检查和设置二进制位

[类型转换](http://www.cnblogs.com/xzxl/p/7655502.html)：显示转换（static\_cast）

术语表：复合表达式、整型提升、短路求值

第五章 语句

表达式语句：在一个表达式的末尾加上分号

switch语句：对括号里的表达式求值，然后和各case标签的值比较

do while语句：先执行后判断；条件为假终止循环

goto语句：容易引发错误，慎用

try语句块：throw表达式、catch子句、异常类

术语表：块、带标签语句、异常声明、terminate

第六章 函数

形参名是可选的：无法使用未命名的形参

局部对象：自动对象、局部静态对象

引用形参：是对应实参的别名，可避免拷贝

const形参：p190

数组引用形参：形参是数组的引用，有限制（大小固定）

main的形参的使用：处理命令行选项

[可变形参](http://www.cnblogs.com/xzxl/p/7660016.html)：initializer\_list类型的形参、省略符形参

返回值：初始化调用点的临时量

引用返回左值：返回引用的函数

返回花括号包围的值列表

主函数main的返回值

返回数组指针使用的类型别名：尾置返回类型、decltype

重载、内联函数、默认实参

constexpr函数：返回常量表达式的函数，被隐式地声明为内联函数

[调试帮助](http://www.cnblogs.com/xzxl/p/7619912.html)：assert、NODEBUG

函数指针

术语表：二义性调用、assert、候选函数、可行函数

第七章 类

使用类定义自己的数据类型

this：一个（指向类类型非常量版本的）常量指针，总是指向调用对象

const成员函数：修改this指针的类型（把this设置为指向常量的指针），故该函数不能改变调用它的对象的内容

定义/初始化/拷贝/赋值/销毁 类的对象

构造函数：类控制其对象的初始化过程的方式

合成的默认构造函数：如果不存在类内的初始值，则默认初始化该成员

构造函数初始值列表：后执行函数体

拷贝对象：初始化、传值调用、返回一个对象

赋值操作：使用了赋值运算符时

我们不定义上面这些操作（初始化/拷贝/赋值/销毁），编译器将替我们合成它们

封装性：使用访问说明符，使用户不可访问类的部分成员，public成员定义类的接口，private部分封装了类的实现细节

友元：访问相关联的类的私有成员，友元类的成员函数也可访问

成员函数&内联：定义在类内部的成员函数是自动inline的，而在类外定义的成员函数必须用inline关键字修饰

可变数据成员：mutable关键字声明；即使它是const对象的成员也是可变的

基于const的重载：对某个对象调用重载成员函数时，将根据该对象是否是const决定调用哪个版本

不完全类型：前向声明的类（仅声明类），不能定义以不完全类型作为参数或返回类型的函数

委托构造函数

[explicit](http://www.cnblogs.com/xzxl/p/7755152.html)：转换构造函数、一步类类型转换

聚合类、字面值常量类

类的静态成员：不能在类的内部初始化静态成员，但可以为constexpr的静态成员提供const整型的类内初始值

第二部分 C++标准库

第八章 IO库

通过一族定义在标准库中的类型来处理IO

IO类：istream/ostream/iostream、ifstream/ofstream/fstream、istringstream/ostringstream/

Stringstream

I O类型间的关系：类型ifstream和istringstream都继承自istream，可将派生类对象当作其基类对象使用

IO对象无拷贝或赋值：不能拷贝或对IO对象赋值，即形参或返回类型必须是引用类型

IO库条件状态：可被任何流类使用的一组标志和函数，指出给定流是否可用

输出缓冲区：保存程序读写的数据，可以管理该缓冲区

文件输入输出：绑定文件的方式、按指定mode打开文件、关闭关联文件close

文件模式：类fstream定义的一组标志，在打开文件时指定，用来控制文件如何被使用

术语表：字符串流、stringstream

第九章 顺序容器

类型：vector、deque、list、forward\_list、array、string

array：一种数组类型，其对象大小是固定的

使用元素类型的默认构造函数：只接受容器大小参数时，类必须要有默认构造函数

容器操作：类型别名，支持<、<=运算符，所有容器都支持相等运算符

emplace操作：构造而非拷贝元素

使迭代器失效的容器操作：p315

容器适配器：为容器操作定义了不同的接口，来与容器类型适配

术语表：适配器、左闭合区间

第十章 泛型算法

标准库容器定义了很少的操作（添加、删除元素，访问首尾元素等），有更多的操作（如查找特定元素、重排元素等）需要通过算法实现。

迭代器令算法不依赖于容器，但算法依赖于元素类型的操作

[算法](http://www.cnblogs.com/xzxl/p/7694706.html)：find、count、accumulate、equal、fill、replace、sort、unique、stable\_sort、for\_each、transform

插入迭代器：一种向容器中添加元素的迭代器，back\_iterator

谓词作为参数：谓词是一个可调用的表达式，其返回结果是一个能用作条件的值，如cmp（返回能作为条件的bool值，能调用cmp(a, b)）

标准库算法或使用一元谓词或使用二元谓词，一元谓词意味着该谓词只能接受一个参数

[lambda表达式](http://www.cnblogs.com/xzxl/p/7818352.html)：当算法要求使用X元谓词，而实际使用到的谓词接受的参数个数大于X时，我们需要使用lambda表达式

可调用对象：可对其使用调用运算符的对象，如函数、函数指针、lambda表达式、重载了函数调用运算符的类

bind函数：解决谓词参数数目问题

插入迭代器：back\_inserter、front\_inserter、inserter

iostream迭代器：将对应的流当做一个特定类型的元素序列来处理

反向迭代器：需要递减运算符

\_copy版本的算法：将元素写到一个指定的输出目的位置

\_if版本的算法：接受一个谓词

链表类型list和forward\_list优先使用成员函数形式的算法：不使用通用算法，因为它们不支持随机访问迭代器

术语表：istream\_iterator、ostream\_iterator、迭代器类别

第十一章 关联容器

元素按关键字来保存和访问

类型：map、set、multimap、multiset、及其相应的无需版本（前面加unordered\_）

map：关键字-值对的集合，关联数组

set：支持高效的关键字查询操作

pair类型：map中的元素的类型，first成员保存关键字，second成员保存对应的值

multi：允许多个元素具有相同的关键字

关联容器不支持顺序容器的位置相关的操作（如push\_back等），因为其中的元素是根据关键字存储的

关键字类型的要求：该类型要求定义了“行为正常”的<运算符

自定义组织一个容器中元素的操作类型：multiset<Sales\_data, decltype(compareIsbn)\*> bookstore(&compareIsbn);

关联容器额外的类型别名：key\_type、mapped\_type、value\_type

添加元素：insert、emplace

访问元素：find(k)、count(k)、low\_bound(k)、upper\_bound(k)、equal\_range(k)

无序容器：组织元素不是通过使用比较运算符，而是使用一个哈希函数和关键字类型的==运算符

术语表：hash、哈希函数、严格弱序

第十二章 动态内存

标准库定义了两个智能指针类型来管理动态分配的对象

每个程序拥有一个内存池，我们称之为自由空间（或堆），程序用堆来存储动态分配的对象

动态分配的对象：在程序运行时分配的对象，其生存期由程序来控制

管理动态内存：new在动态内存中为对象分配空间并返回一个指向该对象的指针（我们可以选择对对象进行初始化）；delete接受一个动态对象的指针，销毁该对象并释放与之关联的内存

使用动态内存易出现的问题：内存泄漏、产生引用非法内存的指针

为了更容易地使用动态内存，新的标准库提供了两种智能指针类型来管理动态对象

[智能指针](http://www.cnblogs.com/xzxl/p/7735582.html)：负责自动释放所指向的对象

shared\_ptr类：初始化/赋值、支持的操作

make\_shared函数：创建一个动态对象并初始化，返回指向该动态对象的shared\_ptr

make\_shared用其参数来构造给定类型的对象，如调用make\_shared<string>时传递的参数必须与string的某个构造函数相匹配

shared\_ptr的析构函数会递减它指向的对象的引用计数，如果引用计数变为0，则还会销毁对象并释放它占用的内存

即如果有n个shared\_ptr指向同一个对象，那该对象的引用计数就是n，于是每销毁一个shared\_ptr时，引用计数就减1

几种递增/递减引用计数的情况：譬如返回一个智能指针将递增计数，因为在调用点会有一个shared\_ptr保存返回来的智能指针

程序使用动态内存的一个原因：程序需要在多个对象间共享数据

Blob类：Blob对象的不同拷贝之间共享相同的元素，[实现代码](http://www.cnblogs.com/xzxl/p/7706088.html" \t "_blank)

[直接管理内存](http://www.cnblogs.com/xzxl/p/7738152.html)：new无法为其动态分配的对象命名，值初始化与默认初始化的区别

new分配的对象都是默认初始化的

通过内置指针管理的动态对象的生存期：直到被显式释放之前，局部指针变量离开作用域会被销毁

delete之后重置指针值：防止空悬指针

shared\_ptr与new结合使用：将一个智能指针绑定到一个用new动态分配的内存上，即用new返回的指针来初始化智能指针

用来初始化智能指针的普通指针：默认是指向动态内存的，如果不是，则必须提供自己的操作来替代delete

不要混用智能指针和内置指针：智能指针的特性只在智能指针之间操作才会有效

unique\_str类：绑定到一个new返回的指针上，不支持普通的拷贝和赋值操作

不能拷贝unique\_str的例外：可以拷贝和赋值一个将要销毁的unique\_ptr

向unique\_ptr传递删除器：默认使用delete释放它指向的对象，可重载一个删除器；类似于重载关联容器的比较操作

weak\_ptr类：不可控制所指向对象生存期的智能指针，指向由一个shared\_ptr管理的对象

StrBlobPtr类：相当于指向StrBlob类的指针，可访问修改vector<string>的元素，[实例代码](http://www.cnblogs.com/xzxl/p/7706088.html" \t "_blank)

动态数组：一次性为很多元素分配内存

[allocator类](http://www.cnblogs.com/xzxl/p/7750887.html)：允许我们将分配和初始化分离，而new将内存分配和对象构造组合在了一起

allocator支持的操作：分配/释放内存、构造/销毁对象

标准库为allocator类定义了两个伴随算法：在分配的原始内存中创建对象

[文本查询程序](http://www.cnblogs.com/xzxl/p/7853583.html)：标准库相关内容学习的总结

术语表：释放器、定位new、引用计数

第三部分 类设计者的工具

第十三章 拷贝控制

类对象的几个操作：拷贝、移动、赋值、销毁

五种特殊函数：拷贝构造函数、拷贝赋值运算符、移动构造函数、移动赋值函数、析构函数

通过定义拷贝控制成员：使类的行为看起来像一个值，或像一个指针

行为像值的类：对于类管理的资源，每个对象都拥有一份自己的拷贝

行为像指针的类：多个对象共同管理相同的资源

拷贝并交换：将左侧运算对象与右侧运算对象的一个副本进行交换，而返回左侧对象，常用于定义赋值运算符

拷贝控制示例：练习应用拷贝控制成员

动态内存管理类：该类在运行时分配可变大小的内存空间，自己进行内存分配，如标准库中的vector类

StrVec类：vector类的简化版本，练习如何实现动态内存管理类，主要知识点为allocator类的应用

对象移动：移动而非拷贝对象，类似IO类、unique\_ptr这样的类的对象不能拷贝但可以移动

右值引用：必须绑定到右值的引用

标准库move函数：可显式地将一个左值转换为对应的右值引用类型，即可以通过它来获得绑定到左值上的右值引用

移动构造/赋值函数：直接接管目标对象的资源，而不需要拷贝

挑选移动还是拷贝函数：根据传递的实参，如实参是右值，当然选移动

noexcept：声明函数不会抛出异常

引用限定符：&和&&放在参数列表后面，分别指出this可以指向一个左值（即只能向可修改的左值赋值）或右值

术语表：拷贝初始化、删除的函数、move、移动迭代器

第十四章 重载运算符与类型转换

不可重载的运算符：:: 、.\* 、. 、?:

作为成员还是非成员：非成员的可以是算术、相等性、关系和位运算，成员的有=/[]/()/->/+=/++/解引用

重载输入运算符>>：需要检查读取操作是否成功，即必须处理输入可能失败的情况

定义前置/后置版本的递增/减运算符：为了区别开来，会使后置版本多一个形式上的整型形参

函数调用运算符：可以像使用函数一样使用该类的对象，与可调用对象的共性，funtion类型

类类型转换：类型转换运算符及使用它应避免二义性

术语表：调用形式、类类型转换、类型转换运算符、函数表、函数对象

第十五章 面向对象程序设计

面向对象程序设计基于三个基本概念：数据抽象、继承和动态绑定

OOP概述：数据抽象（将类的接口与实现分离）、继承（定义相似的类型并对其相似关系建模）、动态绑定（在一定程度上忽略相似类型的区别，而以统一的方式使用它们的对象）

虚函数：某些成员函数，基类希望它的派生类各自定义适合自身的版本

动态绑定（函数调用）：使用基类的引用或指针调用一个虚函数时发生动态绑定

派生类可以继承定义在基类中的成员，但是派生类的成员函数不一定有权访问从基类继承而来的成员

派生类对象的组成：一个含有派生类自己定义的非static成员的子对象 + 一个与该派生类继承的基类对应的子对象（如果由多个基类，此子对象也有多个）

派生类向基类的类型转换：因为派生类对象中含有与其基类对应的组成部分，所以能把派生类的对象当成基类对象来使用，而我们也能将基类的引用或指针绑定到派生类对象中的基类部分上；编译器会隐式地执行派生类到基类的转换！！但该转换只对指针或引用类型有效

类型转换带来的：可以把派生类对象或派生类对象的引用用在需要基类引用的地方，同样也可以把派生类对象的指针用在需要基类指针的地方

派生类不能直接初始化其对象中的基类部分：和其他创建了基类对象的代码一样，必须使用基类的构造函数来初始化它的基类部分

即每个类控制它自己的成员初始化过程

派生类的作用域嵌套在基类的作用域之内：派生类成员可以像使用本类成员一样使用基类的成员

防止继承的发生：在类名之后跟一个关键字final

静态类型&动态类型

抽象基类：含有（或未经覆盖直接继承）纯虚函数的类，我们不能创建抽象基类的对象

派生类访问基类的成员：protected成员的特性

三种继承方式对于访问控制的影响：派生类对其继承而来的成员的访问权限

改变个别成员的可访问性：使用using声明来改变派生类继承的某个名字的访问级别

继承中的名字查找：派生类的作用域嵌套在基类中，对象的静态类型决定了从哪个类开始搜索名字

名字查找先于类型检查：因为派生类的作用域嵌套在基类中，故其中与基类同名的函数会隐藏基类中的函数，即只要找到此名字的函数就停止向基类搜索

虚析构函数：动态分配继承体系中的对象，在基类中将析构函数定义成虚函数

继承与拷贝控制：位于继承体系中的类进行拷贝控制操作时应注意基类的拷贝控制

继承的构造函数：派生类可使用using声明语句继承基类的构造函数，但不能继承默认、拷贝和移动构造函数

容器与继承：使用容器存放继承体系中的对象

文本查询程序再探：扩展文本查询程序，增加更多查询操作，作为继承的最后一个例子

术语表：可访问的、派生类向基类的类型转换、动态类型、覆盖、多态性、重构、公有继承

第十六章 模板与泛型编程

模板是泛型编程的基础

模板的编写格式：函数/类模板、类模板的友元、成员模板

模板实参的深入：类型转换、显式实参

重载与模板：特例化版本优先、非函数模板优先

引用折叠和右值引用参数：函数参数是右值引用

可变参数模板：sizeof...运算符、参数包、包扩展

模板特例化：模板的一个特例化版本，并非重载模板

术语表：模板参数、模板参数列表、类型参数、实例化、成员模板、参数包、函数参数包、类型转换

第四部分 高级主题

第十七章 标准库特殊设施

介绍四个具有特殊目的的标准库设施，以及IO库中某些不常用的部分

tuple类型：类似pair，但包含的类型更多

biset类型：相对位运算来说，处理二进制数更容易

正则表达式：很好的描述字符序列的方法

随机数：随机数引擎+分布类型，我们说的随机数发生器就是指分布对象和引擎对象的组合

IO库再探：一系列操纵符控制输入输出格式、未格式化的输入/输出操作允许将一个流当做一个无解释的字节序列来处理、定位流中位置并随机访问

术语表：tupe、biset、regex、cmatch、smatch、未格式化IO、随机数引擎、随机数分布、操纵符、种子

第十八章 用于大型程序的工具

介绍在设计大型程序时最有用的三个特效，包含异常处理、命名空间和多重继承

noexcept异常说明：指定某个函数不会抛出异常

noexcept运算符：可与noexcept说明符混合使用

异常说明与指针、虚函数和拷贝控制：函数的异常说明对于使用函数的影响

异常类层次：标准库异常类构成了一套继承体系

命名空间：作用域 + using声明 + using指示

命名空间中的名字&多个文件：using声明的作用域+using指示的弊端

重载与命名空间：命名空间对函数的匹配过程的影响

多重继承：从多个直接基类产生派生类

多重继承的初始化过程及其拷贝控制

虚继承：每个派生类最多一次继承同一个类

先初始化虚基类部分，再构造其他非虚基类

术语表：构造函数顺序、文件中的静态声明、全局命名空间、命名空间污染、重新抛出

第十九章 特殊工具与技术

介绍几种用于特定类别问题的特殊工具和技术

重载new和delete：自定义内存分配的细节

使用定位new形式构造对象：传递一个地址，在一个特定的、预先分配的内存地址上构造对象，而不分配内存

运行时类型识别：typeid运算符 + dynamic\_cast运算符

枚举类型：定义新的类型

类成员指针：数据成员指针 + 成员函数指针

嵌套类：作用域 + 名字查找 + 与外层类的关系

union类型：任意时刻只有一个数据成员有值

局部类：类定义在某个函数内部 + 名字查找

固有的不可移植的特性：为了支持低层编程而定义的，不可移植的特性是指因机器而异的特性，包括位域、volatile限定符和链接指示

术语表：匿名union、判别式、RTTI、定位new表达式、枚举类型、dynamic\_cast、链接指示

附录

静态成员 p268

析构函数：销毁元素，释放内存

类型别名：typedef int arrT[10]; arrT是一个由10个整型元素组成的数组的别名

与类的对象交互必须使用该类的接口

通过作用域运算符来使用被隐藏的名字

Sales\_data类：p240

Screen类：p243

StrBlob类：p405

StrBlobPtr类：p421

文本查询程序：p430 T13.42

Message类：p461

StrVec类：p465 T13.44